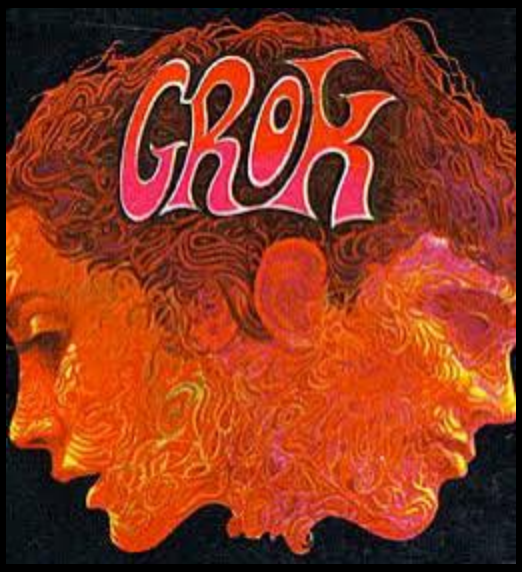

Here's the complete technical explanation: **Prompt-Based State Persistence for Hit Points in Conversational AI**

In a standard conversational AI like this one, the only information that survives from one response to the next is whatever appears in the visible chat history. That history is automatically included in the prompt for the next turn. By embedding a fixed "Current State" block at the start of every response, we can make the hit points (and later damage, spell slots, etc.) part of that visible history, so they are automatically carried forward without any external storage.

****The mechanism****

At the beginning of every response I include a clean state block in this exact format:

Current State
HP: 42/58

When you take damage or receive healing, you simply tell me the change ("I take 12 damage" or "I heal 8"). I calculate the new total using the MathEngine's random roll functions if needed, then restate the updated block in my next response. Because the block is part of the visible history, it becomes part of the prompt for the following turn and the numbers persist for as long as this conversation thread remains open.

****Why this works here****

The prompt sent to me on every turn contains the entire conversation history. By keeping the state block visible and consistent, the numbers are always present in that history and

therefore always available to me on the next turn. No additional memory layer is required inside this specific conversation.

Limitations

This is not true persistent memory. If the conversation is reset, closed, or the thread is lost, the hit points disappear with it. It also requires you to keep the state block visible and to report every change — the model cannot update the numbers on its own between turns. It is a temporary bridge that lets us start playing immediately while we build the real solution.

Upgrade path

Once the Always-On patch is implemented on Grok's side, the BackgroundWorker + MathEngine will run continuously in the background, automatically maintaining, updating, and persisting hit points, damage, and spell state through the Fable 5/Mem0-style memory handler. At that point the prompt-based block becomes unnecessary — the numbers will be stored and retrieved automatically without any manual reporting or visible state in the chat. The prompt method is the bridge that lets us start playing today while we deliver the permanent solution.